จเร เลิศสุดวิชัย

SOFT MAIL

พัฒนาโปรแกรมยูทิลิตี้ด้วย เทอร์โบซี

ตอนที่ 4

นำเสนอโปรแกรมยูทิลิตี้ที่เกี่ยวกับไฟล์ข้อมูลของดีเบส รวม 4 โปรแกรม

โปรแกรมยูทิลิตี้ของมุมภาษาซีในฉบับนี้จะเป็นโปรแกรมยูทิลิตี้ที่เกี่ยวกับไฟล์ข้อมูลของโปรแกรมดีเบส ซึ่งประกอบด้วย 4 โปรแกรมคือ โปรแกรมบีบอัดข้อมูลไฟล์ข้อมูลดีเบส โปรแกรมขยายข้อมูลจากไฟล์ที่ถูกบีบอัดด้วยโปรแกรมแรก การบีบอัดข้อมูลในโปรแกรมยูทิลิตี้จะใช้เทคนิคแบบง่าย ๆ ซึ่งสามารถลดขนาดไฟล์ลงได้พอสมควร ส่วนโปรแกรมที่ 3 เป็นโปรแกรมที่ใช้สำหรับอ่านดูโครงสร้างของไฟล์ข้อมูลดีเบสโดยตรง และโปรแกรมสุดท้ายเป็นโปรแกรมการอ่านข้อมูลจากไฟล์ดีเบสมาแสดงบนจอภาพทีละเรคอร์ด โปรแกรมทั้ง 4 มีรายละเอียดดังนี้

โปรแกรม CDBF

Compress dBase File

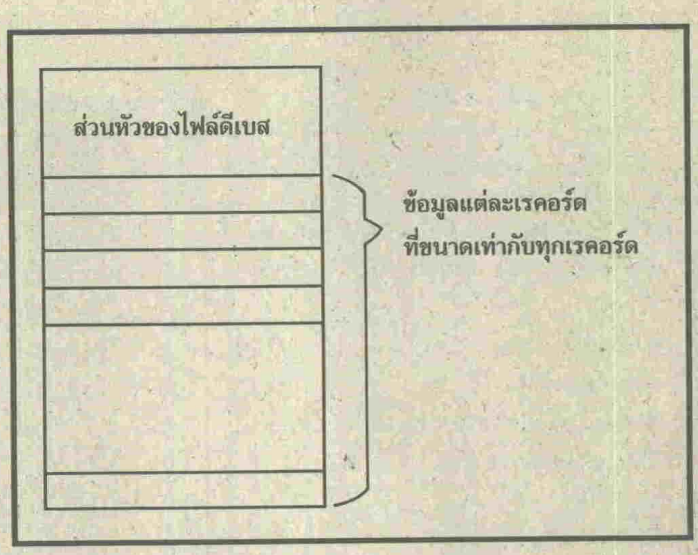
การทำงาน

การเก็บข้อมูลของโปรแกรมดีเบส (และโปรแกรมอื่นที่ใช้โครงสร้างไฟล์เก็บข้อมูลแบบเดียวกัน) มีลักษณะเป็นเรคอร์ด โดยแต่ละเรคอร์ด (Record) จะประกอบด้วยฟิลด์ (Field) ซึ่งถ้าเปรียบเทียบกับตารางข้อมูลเรคอร์ดก็คือข้อมูลในแต่ละแถวของตาราง ส่วนฟิลด์ก็คือข้อมูลในแนวคอลัมน์นั่นเอง และข้อมูลที่จัดเก็บลักษณะนี้จะมีขนาดของฟิลด์และเรคอร์ดคงที่ (Fixed length) การออกแบบสร้างไฟล์ข้อมูลหรือตารางนี้ผู้สร้างจะต้องกำหนดความกว้างของฟิลด์เอาไว้ให้พอที่จะเก็บข้อมูลที่ยาวที่สุด แต่เมื่อใช้งานจริงจะพบว่าข้อมูลที่จัดเก็บในตารางส่วนใหญ่แล้วก็จะมีความยาวข้อมูลไม่เต็มฟิลด์ จึงทำให้การเก็บข้อมูลด้วยไฟล์ข้อมูลดีเบสนี้สิ้นเปลืองเนื้อที่จำนวนมาก

ช่องว่างที่เหลือจากกรณีที่ข้อมูลไม่เต็มฟิลด์ ดีเบสจะเก็บเป็น space หรือแอสกี 32 ซึ่งการบีบอัดข้อมูลของโปรแกรม CDBF จะทำงานโดยการเข้ารหัสของตัวอักษรที่เป็นช่องว่าง (space) โดยใช้วิธีรันเลงธ์ (Run Length)

โปรแกรมนี้สามารถใช้กับไฟล์ข้อมูลลงดีเบสทรี ดีเบสทรีพลัส คลิป-

เปอร์ และฟ็อกซ์เบส (Foxbase) ความสามารถในการบีบอัดข้อมูลได้ถึง 25 เปอร์เซ็นต์ขึ้นไป (บางไฟล์อาจได้ถึง 75 เปอร์เซ็นต์) และไฟล์ผลลัพธ์จะมีนามสกุล .cdb ซึ่งสามารถจัดเก็บแทนไฟล์ .dbf เดิมได้ ประหยัดพื้นที่ดิสก์มากกว่า โดยอาจจะทำการลบไฟล์ .dbf ทิ้ง และเมื่อต้องการใช้งาน .dbf ก็จะขยายข้อมูลคืนจากไฟล์ .cdb ได้ด้วยโปรแกรม EDBF ซึ่งเป็นโปรแกรมยูทิลิตี้ชุดเดียวกัน



รูปที่ 1
แสดงการจัดเก็บไฟล์ข้อมูลของดีเบส ซึ่งเก็บแบบความยาว

การใช้งาน

รูปแบบการใช้งานนั้นเรียกชื่อโปรแกรม CDBF แล้วตามด้วยชื่อไฟล์ดีเบสดังนี้

```
cdbf ชื่อไฟล์ดีเบสที่ต้องการบีบอัด
```

โดยชื่อไฟล์ดีเบสที่ระบุไม่ต้องใส่นามสกุล .dbf โปรแกรมจะจัดการให้เอง

ตัวอย่างที่ 1

```
c: \ >cdbf c: \ clipper \ data \ customer
CDBF-Compress dBase File, TC Editon
```

เป็นการใช้โปรแกรม cdbf บีบอัดข้อมูลไฟล์ customer.dbf ซึ่งอยู่ในไดเรคทอรี c: \ clipper \ data ซึ่งหลังจากบีบอัดข้อมูลแล้วโปรแกรมจะสร้างไฟล์ customer.cdb ให้

ตัวอย่างที่ 2

```
c: \ cdbf c: \ aLpha \ sample \ data
CDBF-Compress dBase File, TC Editon
Error opening the file. Please check your
spelling
```

ในกรณีที่เป็นการเรียกโปรแกรม CDBF เพื่อจะบีบอัดไฟล์ข้อมูลของโปรแกรมอัลฟา (ซึ่งใช้โครงสร้างไฟล์ข้อมูลแบบเดียวกับไฟล์ข้อมูลดีเบส) แต่ไม่มีชื่อไฟล์ data.dbf อยู่จริง หากการตรวจสอบชื่อไฟล์ของโปรแกรม CDBF แล้วไม่พบไฟล์ข้อมูล data.dbf โปรแกรมก็จะรายงานข้อความดังกล่าวให้ทราบตามตัวอย่าง

ตัวอย่างที่ 3

```
C: \ FOX > cdbf
Usage: c: \ >cdbf filename <dbf assumed>
```

หากเรียกชื่อโปรแกรม cdbf อย่างเดียว โปรแกรมจะแสดงข้อความบอกถึงวิธีการใช้งานให้

เทคนิคโปรแกรม

เทคนิคโปรแกรมการบีบอัดข้อมูลของ CDBF นั้นใช้วิธีการง่าย ๆ โดยอาศัยคุณสมบัติทั่วไปของไฟล์ข้อมูลดีเบส ซึ่งมีที่ว่าง (แอสกี 32) เป็นจำนวนมาก เพราะโครงสร้างไฟล์ข้อมูลของดีเบสเป็นแบบความยาวคงที่ (Fixed Length) ดังได้กล่าวไปแล้วในตอนต้น โดยการบีบอัดข้อมูลจะใช้วิธีแบบ Run Length คือนับจำนวนช่องว่างที่อยู่ติดกัน แล้วแทนช่องว่างเหล่านั้นด้วยแอสกี 26 พร้อมกับจำนวนช่องว่างที่นับได้

โปรแกรม CDBF จะบีบอัดข้อมูลของไฟล์เฉพาะบริเวณเนื้อไฟล์ที่เป็นข้อมูลจริง คือหลังส่วนหัว (Header) ของไฟล์ข้อมูล (ดังรูปที่ 1) ข้อมูลบริเวณส่วนหัวจะไม่ถูกบีบอัดแต่จะถูกเขียนลงไปในไฟล์ผลลัพธ์ตรงๆ ซึ่งเป็นข้อดีของโปรแกรม เนื่องจากส่วนหัวของไฟล์ข้อมูลดีเบสจะบอกถึงโครงสร้างไฟล์ข้อมูล (ดังตารางที่ 1 และ 2) ซึ่งทำให้ไฟล์ผลลัพธ์ที่ได้จากการบีบอัด .cdb นั้นยังสามารถบอกถึงโครงสร้างของไฟล์ข้อมูลจริงได้ (ทำให้โปรแกรม DI-dBase File Information สามารถอ่านโครงสร้างไฟล์ .cdb ได้ด้วย)

การทำงานของโปรแกรมจะเริ่มด้วยการเปิดไฟล์ข้อมูลที่ถูกกำหนดมาเมื่อถูกเรียกใช้งาน ซึ่งหลังจากเปิดไฟล์แล้วจะทำการตรวจสอบ หากไฟล์ที่ระบุไม่มีจริงก็จะรายงานข้อผิดพลาดบนจอภาพและส่งรหัสผิดพลาดไปที่ดอสด้วย หากการเปิดไฟล์สำเร็จโปรแกรมจะอ่านข้อมูล

ตารางที่ 1 รายละเอียดข้อมูลส่วนหัวของไฟล์ดีเบส

ไบต์ที่	ขนาด	ความหมายของข้อมูล
0	1 ไบต์	ชนิดของดีเบส 02H-ดีเบสทู 03H-ดีเบสทรี ไม่มีไฟล์ dbf 83H-ดีเบสทรีมีไฟล์ dbt ด้วย
1-3	3 บิต	วันที่แก้ไขล่าสุด
4-7	32 บิต	จำนวนเรคอร์ด
8-9	16 บิต	จำนวนไบต์ในส่วนหัว (header)
10-11	16 บิต	จำนวนไบต์ต่อเรคอร์ด
12-14	3 ไบต์	สงวนได้
15-27	13 ไบต์	ใช้กรณี multiuser
28-31	20 ไบต์	สงวนไว้
32-n	ทีละ 32 ไบต์	เป็นรายละเอียดของแต่ละฟิลด์ (ดูรายละเอียดตารางที่ 2)
n+1	1 ไบต์	ODH เป็นตัวจบฟิลด์

ตารางที่ 2 รายละเอียดของฟิลด์

ไบต์ที่	ขนาด	ความหมาย
0-10	11 ไบต์	ชื่อฟิลด์
11	13 ไบต์	ชนิดของไฟล์ (C,N,L,D,M)
12-15	32 บิต	แอดเดรสฟิลด์ (ใช้ในหน่วยความจำ)
16	1 ไบต์	ความยาวฟิลด์
17	2 ไบต์	ฟิลด์ทศนิยม
18-19	1 ไบต์	สงวนได้
20	2 ไบต์	พื้นที่ ID
21-12	1 ไบต์	ใช้สำหรับ multiuser
23	8 ไบต์	SET FIELD FLAG
24-31	1 ไบต์	สงวนไว้

ไบต์แรกมาตรวจสอบว่าไฟล์ดังกล่าวเป็นไฟล์ข้อมูลของดีเบสหรือเป็นไฟล์พวกเดียวกันหรือไม่ ซึ่งหากไม่ใช่ไฟล์ของดีเบส โปรแกรมจะรายงานข้อผิดพลาดบนจอภาพและที่ดอสเช่นเดียวกัน จากนั้นโปรแกรมจะอ่านเอาเนื้อไฟล์ข้อมูลส่วนหัวออกมาแล้วนำลงไฟล์ผลลัพธ์โดยตรง (ขนาดของความยาวส่วนหัวหาได้จากไบต์ที่ 8 และ 9 ของข้อมูลในไฟล์นับจากต้นไฟล์) จากนั้นโปรแกรมหลัก (main) จะเรียกไปยังฟังก์ชัน compress( ) ซึ่งภายในฟังก์ชันจะอ่านเนื้อข้อมูลของไฟล์มายังบัฟเฟอร์ทีละ 32 กิโลไบต์ แล้วสแกนหาช่องว่างแอสกี 32 ซึ่งหากพบแอสกี 32 ติดกัน โปรแกรมจะนับจำนวนช่องว่าง (ทำในฟังก์ชัน elimspaces) และแทนช่องว่างติดกันเหล่านั้นด้วยแอสกี 26 ตามด้วยจำนวนช่องว่างที่นับได้ จากนั้นจึงเขียนลงไฟล์ผลลัพธ์ ทำเช่นนี้จนจบไฟล์

การบีบอัดข้อมูลโดยวิธีนี้ผู้อ่านอาจนำไปดัดแปลงใช้กับไฟล์ข้อมูลอื่นก็ได้ แต่จะลดขนาดข้อมูลได้ดีกับไฟล์ข้อมูลที่เป็นไฟล์ข้อความ (Text File) สำหรับไฟล์ชนิดอื่น ๆ อาจจะลดขนาดได้ไม่มากนัก

โปรแกรม EDBF

Expand dBase File

การทำงาน

โปรแกรมนี้ทำหน้าที่ขยายไฟล์ข้อมูลนามสกุล .cdb ที่ถูกบีบอัดข้อมูลมาจากโปรแกรม CDBF ซึ่งหลังจากการทำงานของโปรแกรมก็จะสร้างไฟล์นามสกุล .dbf กลับคืนมา

การใช้งาน

รูปแบบการใช้งานโปรแกรม EDBF ก็เช่นเดียวกับโปรแกรม CDBF คือ เรียกชื่อโปรแกรม EDBF แล้วตามด้วยชื่อไฟล์ที่ต้องการขยาย ดังนี้

```
edbf ชื่อไฟล์ที่ต้องการขยาย
```

การระบุชื่อไฟล์ที่ต้องการขยายไม่ต้องใส่นามสกุล โปรแกรมจะถือว่าเป็นนามสกุล .cdb

ตัวอย่างที่ 1

```
c: \ >edbf c: \ Clipper \ data \ customer

EDBF-Expand dBase File, TC-Edition
```

เป็นตัวอย่างการขยายไฟล์ customer.cdb ในไดเรคทอรี c: \ clipper \ data ซึ่งถูกบีบอัดมาก่อน หลังจากการทำงานโปรแกรมจะสร้างไฟล์ข้อมูล customer.dbf กลับคืนมา

ตัวอย่างที่ 2

```
c: \ >edbf c: \ alpha \ sample \ data
EDBF-Expand dBase File, TC-Edition
Error opening the file. Please check the spelling
```

ตัวอย่างนี้เป็นการเรียกโปรแกรม EDBF โดยที่ไฟล์ที่ต้องการขยายไม่มีจริง โปรแกรมจะแสดงข้อความผิดพลาดบนจอภาพและส่งคำผิดพลาดให้กับดอสด้วย

ตัวอย่างที่ 3

หากเรียกชื่อโปรแกรมอย่างเดียว โปรแกรมจะแสดงวิธีการใช้งานบนจอ ดังนี้

```
c: \ DBASE \ >edbf
usage: c: \ >edbf filename <.cdb assumed>
```

เทคนิคโปรแกรม

การทำงานโปรแกรม EDBF จะมีการทำงานตรงข้ามกับโปรแกรม CDBF คือการขยายข้อมูลในไฟล์ .cdb โปรแกรมจะสร้างบัฟเฟอร์ 2 ชุด ชุดแรก (inbuffan) สำหรับข้อมูลที่ถูกบีบอัด ซึ่งโปรแกรมจะอ่านข้อมูลจากไฟล์ .cdb มาใส่ในบัฟเฟอร์ จากนั้นโปรแกรมจะทำการตรวจสอบแอสกี 32 ในบัฟเฟอร์ชุดแรก ซึ่งหากพบก็จะทำการขยายช่องว่างให้มีจำนวนเท่ากับจำนวนช่องว่าง ซึ่งเป็นตัวเลขที่อยู่ตามหลังรหัส 32 ข้อมูลที่ถูกขยายและไม่ถูกขยาย (เพราะไม่ได้ถูกย่อมาหรือคือข้อมูลที่ไม่ใช่รหัส 32 จะถูกเขียนลงในบัฟเฟอร์ชุดที่ 2 (outbuffer) และถูกนำเขียนไฟล์ผลลัพธ์ .dbf ในที่สุด บัฟเฟอร์ทั้ง 2 จะมีขนาด 32 กิโลไบต์

สำหรับข้อมูลในส่วนหัวของไฟล์ซึ่งไม่ได้ถูกบีบอัดมาก็จะถูกเขียนลงไฟล์ผลลัพธ์โดยตรงตั้งแต่แรก

โปรแกรม DI.EXE

dBase File Information

การทำงาน

การดูรายละเอียดของไฟล์ข้อมูลดีเบสนั้น ต้องทำผ่านโปรแกรมดีเบสอินเตอร์พรีเตอร์ โดยเรียกโปรแกรมดีเบส แล้วใช้คำสั่งเปิดไฟล์ (USE) และตามด้วยคำสั่งขอดูโครงสร้างไฟล์ (Display Structure) แต่ในบางโอกาสวิธีนี้ก็สร้างความไม่สะดวกให้แก่เราที่ต้องคอยอาศัยการทำงานของโปรแกรมดีเบสหรือบางครั้งก็ไม่สามารถหาโปรแกรมดีเบสได้ โปรแกรม DI จะช่วยการทำงานในส่วนนี้ให้กับเรา ซึ่งโปรแกรม DI มีขนาดเล็ก ทำงานก็ไม่ซับซ้อน จึงทำงานได้เร็วช่วยให้การทำงานสะดวกขึ้น

โปรแกรม DI จะอ่านข้อมูลจากข้อมูลส่วนหน้า (Header File) ของไฟล์ข้อมูลดีเบส แล้วพิมพ์รายละเอียดในเรื่องของโครงสร้างไฟล์ให้บนจอภาพ นอกจากจะสามารถอ่านไฟล์ข้อมูลของดีเบสแล้วยังสามารถอ่านไฟล์ของโปรแกรมอื่นที่มีโครงสร้างเดียวกับโปรแกรมดีเบส เช่น โปรแกรมคลิปเปอร์ ฟ็อกซ์เบส หรือโปรแกรมอัลฟา ส่วนไฟล์ข้อมูลที่ได้รับการบีบอัดมาจากโปรแกรม CDBF นั้น โปรแกรม DI ก็สามารถอ่านรายละเอียดได้ด้วย เพราะตามการทำงานของโปรแกรม CDBF จะไม่บีบอัดข้อมูลส่วนหัวของไฟล์ข้อมูลดีเบส

การใช้งาน

การใช้งานโปรแกรม DI ทำได้ง่ายโดยการเรียกโปรแกรม DI แล้วตามด้วยชื่อไฟล์ข้อมูลดีเบสที่ต้องการดูโครงสร้าง ดังนี้

```
DI ชื่อไฟล์ข้อมูล. นามสกุล
```

การระบุไฟล์ข้อมูลต้องบอกทั้งชื่อและนามสกุลให้ครบ

ตัวอย่างที่ 1

```
c: \ > di C: \ MAGAZINE \ ISSUE.DBF
```

โปรแกรมจะแสดงรายละเอียดโครงสร้างของไฟล์ข้อมูล ISSUE.DBF ดังแสดงในรูปที่ 2

ตัวอย่างที่ 2

```
C: \ >di    C: \ alpha \ data \ customer
DI-dBASE File Information, TC Edition
The file was not found. Please check the spelling
```

```
DI-dBase File Information, TC Edition

Name of dBase File      : ISSUE.DBF
Date of Last update     : Sep 17, 1990
Number of Records       : 22
Header Length           : 194 bytes.
Record Length           : 29 Bytes.
Number of fields        : 5

Field Name      Field Type      Length      Decimals
-----------------------------------------------------
MAG_ID          C               5
VOLUME_ID       C               3
ISSUE_ID        C               2
DATE            D               8
NOTES           M               10
```

รูปที่ 2

ตัวอย่างนี้เป็นเรียกโปรแกรมเพื่อให้ดูรายละเอียดของไฟล์ customer.dbf แต่ไม่ได้ระบุนามสกุลของไฟล์ข้อมูลให้โปรแกรม ซึ่งตามที่ได้กล่าวไปแล้วว่าการเรียกใช้โปรแกรม DI นั้นต้องระบุชื่อและนามสกุลของไฟล์ข้อมูลให้ชัดเจน

ตัวอย่างที่ 3

```
C: \ CLIPPER > di C: \ autoexec.bat
dI-dBase File Information. TC Edition
```

ก่อนจะอ่านข้อมูลจากไฟล์โปรแกรม DI จะต้องตรวจสอบเสียก่อนว่าไฟล์ข้อมูลที่ระบุเป็นไฟล์ข้อมูลที่มีโครงสร้างแบบของดีเบสหรือไม่มิฉะนั้นโปรแกรมจะไม่ทำงานต่อ ดังตัวอย่างนี้เป็นต้น

ตัวอย่างที่ 4

```
C: \ DBASE>di
DI-dBASE File Information, TC Edition
Usage: C: \ > di filename.dbf
```

หากเรียกโปรแกรมอย่างเดียวจะแสดงวิธีใช้งานให้

เทคนิคโปรแกรม

เทคนิคโปรแกรม DI จะอาศัยการสร้างโครงสร้างข้อมูล (Structure) ตามโครงสร้างไฟล์ของดีเบส ซึ่งหากโปรแกรมตรวจสอบแล้วว่าไฟล์ที่ระบุเป็นไฟล์ข้อมูลดีเบส โปรแกรมจะอ่านข้อมูลจากไฟล์ลงยังตัวแปรที่มีโครงสร้างข้อมูลส่วนหัว (ตัวแปร header) สำหรับรายละเอียดของฟิลด์ของข้อมูลก็จะถูกอ่านเข้าไปยังตัวแปรที่มีโครงสร้างของรายละเอียดฟิลด์ (field-info) และพิมพ์ออกมาที่จอภาพ

โครงสร้างข้อมูลของไฟล์ข้อมูลของคลิปเปอร์จะแตกต่างกับดีเบสในเรื่องของตามยาวของฟิลด์ชนิดข้อความ (character) ส่วนไฟล์ข้อมูลฟ็อกซ์เบสจะแตกต่างกับดีเบสที่ความยาวของข้อมูลส่วนหัวจะมากกว่าไฟล์ดีเบสไบต์หนึ่ง ซึ่งสามารถตรวจสอบได้ที่ไบต์สุดท้ายของส่วนหัวจะเป็นแอสกี 13 (carriage return)

โปรแกรม DBLIST

dBase File List

การทำงาน

โปรแกรม DBLIST มีความสามารถในการอ่านข้อมูลจากไฟล์ดีเบสมาแสดงบนจอภาพทีละเรคอร์ด โปรแกรม DBLIST ก็เช่นเดียวกับโปรแกรม DI ที่มีขนาดโปรแกรมเล็กจึงทำงานได้เร็ว ซึ่งจะช่วยให้ผู้ใช้งานสามารถดูข้อมูลจากไฟล์ข้อมูลดีเบสได้โดยตรงไม่ต้องผ่านโปรแกรมดีเบสอินเตอร์พรีเตอร์

การใช้งาน

การใช้งานโปรแกรม DBLIST ก็เช่นเดียวกับโปรแกรม DI คือเรียกชื่อโปรแกรมแล้วตามด้วยชื่อไฟล์ข้อมูลที่ต้องการดูข้อมูล ดังนี้

```
DBLIST ชื่อไฟล์ข้อมูล.นามสกุล
```

ตัวอย่าง

ต้องการดูข้อมูลในไฟล์ INVOICES.DBF ดังนี้ (รูปที่ 3)

```
C: \ >list C: \ clipper \ invioces.dbf
```

เทคนิคโปรแกรม

โปรแกรม DBLIST จะอ่านเอารายละเอียดของไฟล์ข้อมูลดีเบส ซึ่งหมายถึงโครงสร้างข้อมูลแต่ละฟิลด์มาเก็บเอาไว้ในรูปแบบของลิงค์ลิสต์ จาก

DBLIST - dBase File List, TC-Edition

1	CC	010	01	19871201	<Memo>
2	DBA	006	02	19880201	<Memo>
3	DBA	006	03	19880301	<Memo>
4	DBA	006	04	19880401	<Memo>
5	DBA	006	05	19880501	<Memo>
6	DBA	006	06	19880601	<Memo>
7	DBA	006	07	19880701	<Memo>
8	DBA	006	08	19880801	<Memo>
9	DBA	006	09	19880901	<Memo>
10	PC	007	01	19880101	<Memo>
11	PC	007	02	19880115	<Memo>
12	PC	007	03	19880201	<Memo>
13	PC	007	04	19880205	<Memo>
14	DBA	006	12	19880201	<Memo>
15	CC	012	01	19870101	<Memo>
16	DBA	005	09	19880501	<Memo>
17	DBA	005	10	19881001	<Memo>
18	CC	011	01		<Memo>
19	DBA	006	11	19881101	<Memo>
20	DBMS	001	02	19881001	<Memo>
21	DBMS	001	01	19880901	<Memo>
22	DBMS	001	03	19881101	<Memo>

รูปที่ 3

นั้นจึงอ่านข้อมูลทุก ๆ เรคอร์ดมาแล้วพิมพ์ออกทางจอภาพตามโครงสร้างข้อมูลของฟิลด์แต่ละฟิลด์ สาเหตุที่ต้องเก็บรายละเอียดโครงสร้างของแต่ละฟิลด์เป็นลิงค์ลิสต์ เพราะเนื่องจากไฟล์ข้อมูลแต่ละไฟล์จะมีจำนวนฟิลด์ไม่เท่ากัน ขึ้นอยู่กับการออกแบบเพื่อเก็บข้อมูล ดังนั้นจึงไม่สามารถที่จะเตรียมการเก็บโครงสร้างของฟิลด์ได้ล่วงหน้า

การแสดงผลบนจอภาพไม่ได้มีการจัดการเรื่องรูปแบบของข้อมูล รวมทั้งการแสดงผลจะแสดงจนกว่าข้อมูลในไฟล์จะหมด ซึ่งหากผู้อ่านต้องการเปลี่ยนแปลงก็สามารถดัดแปลงโปรแกรมต้นฉบับได้

โปรแกรมต้นฉบับ

โปรแกรมต้นฉบับทั้งสี่หากผู้อ่านต้องการก็หาได้จากซอฟต์เมล์ประจำฉบับครับ

```
typedef struct
{
  unsigned char dbf_id;
  char last_update[3];
  long last_rec;
  unsigned data_offset;
  unsigned rec_size;
  char filler[20];
} DBF_HEADER;

typedef struct
{
  char field_name[11];
  char field_type;
  char dummy[4];
  union
    {
    unsigned char_len;
    char date_len;
    struct
      {
      char len;
      char dec;
      } num_size;
    } len_info;
    char filler[14];
} FIELD_INFO;

typedef struct field_flage
{
  FIELD_INFO field_info;
  struct field_flage *next_flist;
} FIELD_LIST;

#define DB_WITHOUT_MEMO  0x03
#define DB_WITH_MEMO     0x83
```

รูปที่ 4 ไฟล์ DBF.Hประกาศโครงสร้างของไฟล์ข้อมูลดีเบส

```
/*
  Title          :  CDBF.EXE - Compress dBase files.
  Purpose        :  Will compress dBase III or compatible file by
                    replaces redundant space characters with a code
                    followed by count.
  Requirements   :  The program will work on MS-DOS and PC-DOS
                    computer using version 2.0 or later
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dos.h>
#include <dir.h>
#include <io.h>
#include "dbf.h"

/*
Internal routines used in this uutility.
*/

void compress (void);
int elimspaces (int buffilled);
```

รูปที่ 5 โปรแกรมต้นฉบับ CDBF.C

รูปที่ 5 (ต่อ)

```
/*
Global variables used throughout this program.
*/

FILE *dbfile, *cdbfile;
DBF_HEADER header;
char infile[MAXPATH], outfile[MAXPATH];
int  inhandle, outhandle, buffiled; unsigned int
i;
unsigned long numofrecs;
unsigned char *buffer;

/*
This main function sets up a buffer and tests wether the memory allocation succeded. The command line
arguments are then fetched and used as filenames for opening the input and output files.

        CALLED FROM: (DOS)
CALLS:      compress
*/
main (int argc, char *argv[])
{
        printf ("\nCDBF-Compress dBase File, TC Edition\n");

                buffer = (unsigned char *) calloc(32767,1);
        if (argc > 1)
{
        strcpy (infile,argv[1]);
        strtok(infile,".");
        strcpy (outfile,infile);

        strcat (infile,".dbf");
        strcat (outfile,".cdb");
        }
else
        {
        printf ("\nUsage: C:\>compress filename <dbf assumed >\n");
        return (1);
        }

dbfile  = fopen (infile,  "rb");      /* Open files for reading. */
                        cdbfile = fopen (outfile, "wb");      /* Open files for writing. */

if (dbfile == NULL)
        {
        printf ("\nError opening the file. Please check your spelling.\n"); return (2);
        }

inhandle  = fileno (dbfile);
outhandle = fileno (cdbfile);

read (inhandle, (char *)&header, sizeof (DBF_HEADER));

if ((header.dbf_id != DB_WITHOUT_MEMO) && (header.dbf_id !=
        DB_WITH_MEMO))
        {
        printf ("\nNot a dBase III compatible file.\n"); return (3);
        }

fseek  (dbfile,0,0);
fseek (cdbfile,0,0);
read (inhandle , buffer, header.data_offset);
write (outhandle, buffer,header.data_offset);

if (header.last_rec != 0)
        compress ();
else
        {
        printf ("\nJust a dBase header. Since the file contains no"); printf ("\nrecords, there is nothing to
        compress.\n");
        }

fclose (dbfile);
fclose (cdbfile);

if (header.last_rec == 0)
        unlink (outfile);

free(buffer);

  return (0);
}

/*
This function is used for reading in a buffer of data at a time and
calling eliimspaces to remove the spaces (ASCII 32) from the buffer.
Once this is completed, compress writes out the compressed data to
the desination file and continues the process until all the bytes
in the source file are read and compressed.

CALLED FROM:    main
CALLS:          elimspace

*/

void compress (void)
{
 /*
 Variable specific to COMPRESS.
 */
```

รูปที่ 5 (ต่อ)

```
  long size,numread;
  int numtoget;

  size = filelength (inhandle);
  numread = header.data_offset;

  if (size <= 32767)
    numtoget = size - numread;
  else
    numtoget = 32767;
  do
    {
    fseek (dbfile,numread,0);
    read  (inhandle, buffer, numtoget);
    numread += numtoget;

    write ( outhandle,buffer,elimspaces(numtoget));

    if ((numread + numtoget) > size)
      numtoget = size - numread;
    }
  while (numtoget !=0);

}

/*
This function is called by compress to perform the actual
compression and to return the number of bytes that must be
writeten out to the desination file.

CALLED FROM:     compress
CALLS:           (none)

*/

int elimspaces (int buffilled)
{
 /*
 Variables specific to elimspaces.
 */
 unsigned int j, newbufsize;
 unsigned char count;

 j = 1;
 newbufsize = 1;

 do
   {
  if ((buffer[j] == 32) && (j != buffilled))
    {
    count = 0;

    do
      {
      count++;
      }
    while ((buffer[j + count] == 32) &&
         ((j + count) != buffiled) &&
         (count != 255));
    if (count > 1)
      {
      buffer[newbufsize] = 26;
      buffer[newbufsize + 1] = count;
      newbufsize += 1;
      j = j + count -1;
      }
    else
      buffer[newbufsize] = buffer[j];
    }
  else
    buffer[newbufsize] = buffer[j];
  newbufsize++;
  j++;
  }
 while (j < buffilled);

 return (newbufsize);
}
```

รูปที่ 6 โปรแกรมต้นฉบับ EDBF.C

```
/*
Title          :   EDBF.EXE - Expann a Compressed dBase Files

Purpose        :   Will expand a compressed dBase III or compatible
                   file.

Requirements   :   The program will work on MS-DOS and PC-DOS
                   computer using version 2.0 or later
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dir.h>
#include <io.h>
#include "dbf.h"

/*
Internal routines used in this uutility.
*/

void expand (void);
int expandspaces (int buffilled);

/*
Global variables used throughout this program.
*/

FILE  *cdbfile, *dbfile;
DBF_HEADER header;
char  infile[MAXPATH], outfile[MAXPATH];
long size,numread;
int     inhandle, outhandle, buffiled;
unsigned char *inbuffer, *outbuffer;

/*
This main function sets up two buffers and tests wether the memory allocation succeded. The command line
arguments are then fetched and used as filenames for opening the input and output files.

        CALLED FROM: (DOS)
CALLS:      expand
*/
```

รูปที่ 6 (ต่อ)

```
main (int argc, char *argv[])
{
        printf ("\nEDBF-Expand dBase File, TC Edition\n");

        inbuffer  = (unsigned char *) calloc(32767,1); outbuffer = (unsigned char
        *) calloc(32767,1); if (argc > 1)
                {
                strcpy (infile,argv[1]);
                strtok(infile,".");
                strcpy (outfile,infile);

                strcat (infile,".cdb");
                strcat (outfile,".dbf");
                }
        else
                {
                printf ("\nUsage: C:\>expand filename <cdb assumed >\n");
                return (1);
                }

        cdbfile = fopen (infile, "rb");         /* Open files for reading. */
        dbfile  = fopen (outfile,  "wb");       /* Open files for writing. */

        if (cdbfile == NULL)
                {
                printf ("\nError opening the file. Please check your spelling.\n"); return (1);
                }

        inhandle  = fileno (cdbfile);
        outhandle = fileno (dbfile);

                read (inhandle, (char *)&header, sizeof(DBF_HEADER));

        if ((header.dbf_id != DB_WITHOUT_MEMO) && (header.dbf_id !=
                DB_WITH_MEMO))
                {
                printf ("\nNot a dBase III compatible file.\n"); return (2);
                }


        fseek(cdbfile,0,0);
        fseek(dbfile,0,0);

        read  (inhandle  , inbuffer, header.data_offset); write (outhandle , inbuffer,
        header.data_offset);

                expand ();

                fclose (cdbfile);
                fclose (dbfile);

                free(inbuffer);
                free(outbuffer);
                return (0);
        } /* main end. */

        /*
This function is used for reading in a buffer of data at a time and calling expandspaces to expand all coded
spaces. This function in conjunction with expandsapce reverses the process wich the compress utility performs.

CALLED FROM:    main
CALLS:          expandspace

*/

void expand (void)
{
        /*
        Variable specific to expand
        */


        int numtoget;

        size = filelength (inhandle) +1;
        numread = header.data_offset;

        if (size <= 32767)
                numtoget = size - numread;
        else
                numtoget = 32767;
        do
                {
```

รูปที่ 6 (ต่อ)

```
            fseek (cdbfile,numread,0);
            read  (inhandle, inbuffer, numtoget);

            numread += numtoget;
            write ( outhandle, outbuffer, expandspaces(numtoget));

            if ((numread + numtoget) > size)
                  numtoget = size - numread;
            }
      while (numtoget !=0);

}

/*
This function is called by expand, a buffer full of data is inbuffer. This function takes that data and checks it as it writes it to
outbuffer. Each time that as ASCII 26 is found, the function knows that it must substitue a number of spaces (ASCII 32) for it.
The number wich follows the 26 in the output buffer and file.

CALLED FROM:    compress
CALLS:          (none)

*/

int expandspaces (int buffilled)
{
      /*
      Variables specific to elimspaces.
       */

      unsigned int  newbufsize;
      int count, j;

      j = 0;
      newbufsize = 0;

      do
            {
                  if ((inbuffer[j] == 26) && (j != buffilled) &&
                     (newbufsize + inbuffer[j+1] <= 32767))
                  {
                  for (count = 1; count <= inbuffer[j+1]; count++ ) {
                outbuffer[newbufsize] = 32;
                  newbufsize++;
                  }
                    j += 2;
                    }
            else
                    {
                        outbuffer[newbufsize] = inbuffer[j];
                    newbufsize++;
                    j++;
                    }

            if (newbufsize >= 32767)
                    {
                        write(outhandle, outbuffer, 32767);
                    newbufsize = 0;
                    }
            }
      while (j <= buffilled);

      if (numread == size)
            outbuffer[newbufsize-1] = 0x1A;   /* EOF mark */

      return (newbufsize);
}
```

รูปที่ 7 โปรแกรมต้นฉบับ DI.C

```
/*
 Title          :  DI.EXE  - dBase File Information

 Purpose        :  Provides a means for finding information about a
                   dBasecompatible file with out having to own or
                   use the dBase interpreter, Clipper, Foxbase programs.
                   It has been tested with these compatible products
                   and proven to work properly. Certain lines in the
                   program are included to insure borad based compatibility.

 Requirements   :  The program will work on MS-DOS and PC-DOS
                   computer using version 2.0 or later
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <io.h>
#include "dbf.h"

/*
Global variables used throughout the program. The array called
                months is used for converting a numeric month to three character
of month name. If you want to use the full names of months, remove the comment from around the second array and
replace comment around the first.
*/

FILE *dbfile;
DBF_HEADER header;
FIELD_INFO field_info;
int  handle;
char filestr [80];
unsigned int i, j, numoffields;
unsigned char bmonth, bday, byear,*buffer;

                char *months[] = { "Jan", "Feb", "Mar", "Apr", "May", "Jun",
                                   "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" };
/*
                char *months[] = { "January", "February", "March", "April",
                "May", "June", "July", "August",
                "September", "October", "November", "Deceber" };
*/

/*
This main loop is the entire program. The program could have been designed to call differemt procedures, but it was
so strightforward that iit works quit well as one  l;argmain procedure. As use, the program scanned for argument and

CALLED FROM:  (DOS)
CALLS:        (none)
*/

main (int argc, char *argv[])
{
    printf ("\nDI-dBase File Information, TC Edition\n");

    buffer  = (unsigned char *) calloc (16418,1);
    if (buffer == NULL)
        {
        printf ("\nNot enough memory to run application. Please remove"); printf ("\nall memory resident software and
        try again.\n"); return (1);

        }

    if (argc >1)
        strcpy (filestr,argv[1]);
    else
        {
        printf ("\nUsage:di filename.dbf\n");
        return (2);
        }
```

รูปที่ 7 (ต่อ)

```
                    dbfile = fopen (filestr,"r");   /* Open file for reading. */
if (dbfile == NULL)
    {
    printf ("\nThe file was not found. Please check the spelling.\n"); return (3);
    }

handle = fileno (dbfile);
read (handle, (char *)&header, sizeof(DBF_HEADER));

if ((header.dbf_id != 3) && (header.dbf_id != 131))
    {
    printf ("\nNot a dBase III compatible file. This program was"); printf ("\nspecifically designed for dbase format
    files. It"); printf ("\nhas been tested with FoxBase and Clipper as well.\n"); return (4);


    }

printf ("\nName of dBase File  : %s",strupr(filestr));

byear  = header.last_update[0];
bmonth = header.last_update[1];
bday   = header.last_update[2];
                printf ("\nDate of Last update : %s %d, 19%d",months[bmonth-1],
                 bday, byear);
                    printf ("\nNumber of Records   : %ld", header.last_rec);
printf ("\nHeader Length       : %u bytes.",header.data_offset);

                    if (header.filler[19] == 13)      /* Foxbase specific */
      header.data_offset++;
printf ("\nRecord Length       : %u bytes.", header.rec_size);

numoffields = (header.data_offset - 34) / 32;
printf ("\nNumber of fields    : %u", numoffields);

printf ("\n\nField Name    Field Type     Length     Decimals"); printf (
"\n_____");
for (i = 32; i <= (numoffields*32); i += 32)
    {
                read (handle, (char *)&field_info, sizeof(FIELD_INFO));
      switch (field_info.field_type)
          {
          case 'N':
              printf("\n%-10s      %c        %5d        %2d", field_info.field_name, field_info.field_type,
                          field_info.len_info.num_size.len, field_info.len_info.num_size.dec);


              break;

          case 'D':
          case 'L':
              printf("\n%-10s      %c        %5d", field_info.field_name, field_info.field_type,
                          field_info.len_info.date_len);

              break;

          default:
              printf("\n%-10s      %c        %5d", field_info.field_name, field_info.field_type,
                          field_info.len_info.char_len);

          }
      }
    fclose (dbfile);
    free(buffer);

    return (0);
} /* main end. */
```

รูปที่ 8 โปรแกรมต้นฉบับ DBLIST.C

```
Title          :  DBLIST.EXE  - dBase File List
                  To list data in dBase file with out having to own or
                  use the dBase interpreter, Clipper, Foxbase programs.
                  It has been tested with these compatible products
                  and proven to work properly. Certain lines in the
                  program are included to insure borad based compatibility.

Requirements   :  The program will work on MS-DOS and PC-DOS
                  computer using version 2.0 or later
*/

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <string.h>
#include <io.h>
#include <alloc.h>

#include "dbf.h"

#define TURE   1
#define FASLE  0

FIELD_INFO field_info;
                FIELD_LIST *field_list, *flist_ptr, *last_flist;
DBF_HEADER header;

main (int argc, char * argv[])
{
      int handle, more_fields;
            unsigned num_fields, rec_num, rec_offset, width;
      char *buffer,*record;

      printf ("DBLIST - dBase File List, TC-Edition\n"); /* Check for proper number of
      parameter */
      if (argc != 2)
            {
                  printf ("\nUsage:dblist <filename>\n");
            exit (1);
            }

      handle = open (argv[1], O_RDONLY | O_RDWR | O_BINARY); if (handle == -1)
            {
                  printf ("Error opening %s\n",argv[1]);
            exit (2);
            }

      /* Read header */
      read(handle, (char *)&header, sizeof (DBF_HEADER));

      /* Build dynamic field_list linked list */
      last_flist = NULL;
      more_fields = TURE;
      num_fields = 0;

      /* For each field */
      while (more_fields)
            {
            more_fields = (read(handle,(char *)&field_info, sizeof (FIELD_INFO))
                                    ==
                                    sizeof (FIELD_INFO));
            if (more_fields)
                  {

                  /* Is this the last field ? */
                  more_fields = (field_info.field_name[0] != 0x0d);

                  if (more_fields)
                        {
                        /* add to list */
                        num_fields++;
                        flist_ptr = (FIELD_LIST*)
                              malloc (sizeof (FIELD_LIST));

                        memcpy ((char *) &flist_ptr->field_info, (char *) &field_info,
                                    sizeof (FIELD_LIST));

                        flist_ptr->next_flist = NULL;

                        /* Is this the first entry in the list ?*/ if (last_flist)
                              {
                              /* No, add to list */ last_flist->next_flist = flist_ptr;
                              last_flist = flist_ptr;
                              }
                        else
```

รูปที่ 8 (ต่อ)

```
                        {
                        /* Yes, start list */
                        field_list = flist_ptr;
                        last_flist = field_list;
                        }
                    }
                }
        }
/* Field list built, list records */ flist_ptr =
field_list;

/* Position file to start of data */
        if (lseek(handle, (long) header.data_offset,0) == -11) {
                printf ("Error seeking in database file\n");
        exit (1);
        }

/* Allocate space for 1 record */
record = (char *) malloc (header.rec_size);

/* This caters to record with just 1 field */
buffer = (char *) malloc (header.rec_size);

printf ("\n\n");

/* For each record ... */
for (rec_num = 1; rec_num <= header.last_rec; rec_num++) {
        read (handle, record, header.rec_size);
        printf ("%4d ", rec_num);

        /* For each field */
        flist_ptr = field_list;

        /* Print deleted character */
        printf ("%c ",record [0]);

                rec_offset = 1;
        while (flist_ptr)
                {
                        switch (flist_ptr->field_info.field_type)
                {
                case 'N':
                        width = flist_ptr-> field_info.len_info.num_size.len;
                        strncpy (buffer, (char *) &record[rec_offset], width);
                        break;

        case 'D':
        case 'L':
                width = flist_ptr-> field_info.len_info.date_len;
                strncpy (buffer, (char *) &record[rec_offset], width);
                break;

        case 'M':
                width = strlen ("<Memo>");
                    strcpy (buffer, "<Memo>");
                break;
        default:
                width = flist_ptr-> field_info.len_info.char_len;
                strncpy (buffer, (char *) &record[rec_offset], width);
        }
buffer[width] = '\0';
  printf ("%s ",buffer);
rec_offset = rec_offset + width; flist_ptr =
flist_ptr->next_flist;

                }
                printf ("\n");
                }
}
```